Trends from News Headlines Project

GitHub Repository:- https://github.com/dhathrigundum/Group-11-NLP_news-headlines

Teammates:

Sai Dev Prakash Janapareddi, ID: 11507630

Dhathri Gundum, ID: 11442139

Sri Harsha Swaraj Nadendla, ID:11501903

Introduction: -

- **Motivation: -**Lately, digitalization has been the primary goal of the world, because of which the amount of data in every domain, be it State of art neural networks, Google's search engine data, News data, Cancer data, Tweets data, every possible domain one can imagine has an almost un-maintainable amount of data. One such domain where the data over past decades or a couple of decades ago is humongous in size and which provides the possibility to classify and understand what has happened in the world during a period of time is News data. This has been our main motivation. To try understanding and bringing out observations about the events that took place in the world a decade ago. Well, the time period is nothing specific, but we chose a decade as it provides us with enough data to classify and also not so huge to make the task tedious. This further opened the path to analyze the sentiment of the data, rather than just observations. When we dug deep, many of the headlines in the past, in the name of "catchy" headlines, are almost contextless and ambiguous. This limitation has indirectly led meaningful news to become mere stop words in some cases. So, this limitation we observed has driven us to also try obtaining meaningful and sense-making headlines when not-so-meaningful headlines are given as input. Overall, thanks to Natural Language Processing, we are i)classifying the text in the News dataset obtained over a decade, ii) analysing the sentiment in it, and iii) performing topic modeling for the data.

- **Significance: -** Classification of text(news) and placing it in a category makes a lot of sense to understand the data and observe trends, which further leads to many questions and new ideas. Understanding this trend can be useful in many areas like for example: in stocks analyzing the previous information about a stock, its position in each quarter of a year and predicting the market and its effect from other competitors. The major significance of what we believe is, ambiguous headlines will now make sense. When this kind of research is done or projects are made, all the domains which run on the statistics in the past, based on observations, and these trends benefit a lot.

- **Objective: -** To clean the data obtained and pre-process it to make it easy to use in the code. Classifying the text data using predefined labels using the zero-shot hugging face classifier. Then to bring out the shift in the sentiment of the news data, whether it has been positive or negative or neutral and then topic modeling using LDA to understand which headline belongs to which topics. We have chosen to code the

entire project in python and utilize its libraries which have machine learning and NLP tools. Plotting necessary and insightful graphs for better visualization of the outputs obtained.

- **Advancements:-** If time permits, we are very much excited about somehow, involving speech recognition or text to speech or vice-versa. If random keywords about an event in history are given as input, the exact news headlines that match, the sentiment of the headline, and if the headline does not make complete sense, a meaningful headline that could have been appropriate should be returned as an audio(text to speech). This is a speculated idea, if the implementation of all the other features is done, we will try implementing this one too.

- **Features: -** The key features we would like to focus on are Analysing and classifying the data related to news of over a decade, performing sentiment analysis on the data, and topic modeling.. Along with those a few visualizations represent data that help categorize different domains of data.

Background(Related Work):-

Feature Extraction:-

Feature extraction is one of the key techniques that clears paths for many nlp problems. Since our project involves sentiment analysis, model accuracy tests, and techniques like text summarization, breakage of sentences in english(any language for that matter) is quite necessary. Converting the sentences into individual words or cleaning them based on desired requirements like removal of stop words, removal of punctuation etc. In the previously published works, feature extraction has been done with the help of neural networks and deep learning techniques. The paper we took as reference, BIGRU and CRF techniques are implemented. In our project, in the initial stage, we cleaned the data, cleared stop words in English, and removed punctuation marks.

Related IEEE paper:- https://ieeexplore.ieee.org/abstract/document/8999624

Sentiment Analysis:-

Sentiment analysis is the crux of our project. We have done two types of sentiment analysis on both the uncleaned and cleaned data. One type of sentiment analysis is based on polarity using the textblob python library. The other sentiment analysis is emotional sentiment analysis. Emotional sentiment analysis is trying to identify the core emotion of that particular sentence, or a paragraph or a corpus. We have tried to identify 12 different kinds of emotions. Both of these sentiment analysis really give deep insights of the core emotion and whether the sentence is overall positive, negative or neutral based on already existing state of art cnns in the python libraries being used. We then checked the accuracy of both the works. Since our data set is news headlines dataset and tweets data set, sentiment analysis really provides us with key insights for any kind of further classification and querying of data.

Related links/reference:-

https://www.kaggle.com/ishivinal/tweet-emotions-analysis-using-lstm-glove-roberta

https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/

Text Classification:-

Text classification is deciding on certain target labels, and then deciding for each every singular element, each element in the sense in our case, its each token in the result of tokenization, under which of the target labels it falls under. We performed zero shot text classification with hugging face technique. After cleaning the initial data sets, visualizing them for better understanding, in order to further study the data or perform any nlp or machine learning techniques, classification is very important. In the previous works which are already published, previously, only one-shot text classification has been implemented until recent times. Off late, because of the introduction of BERT in NLP, zero-shot text based classification is being possible. In our project, we have implemented the zero shot based text classification tool for classification after the cleaning and sentiment analysis of the data. This classification helps better yield the further techniques to result accurately. In the initial phase, naive approaches like, populating certain databases manually were also done. And then, with the further advancements of nlp techniques and broader scope of machine learning, and specific advancements like zero and one shot classifications, text classifications have been more insightful. Tokenizing or zeroing certain target labels have led many more major breakthroughs in the machine learning and artificial intelligence field. On a high level overview basis, in our project, text classification is basically done by giving each work in the text dataset a certain score, in similar fashion to probabilistic classification. Based on each word's score, each word or token is mapped to its corresponding target label initially set.

Related links/reference:-

https://arxiv.org/pdf/2103.14465.pdf

https://blog.netcetera.com/zero-shot-text-classification-13c5236edcd3

Topic Modeling and Clustering:-

Topic modeling basically in the previous works includes procedures like basically identifying a set of key words or certain targeted elements that are being quite frequent in the data set so far. The overall procedure works in a similar line to bag of words approach, grouping certain target labels together, implementing probability somehow, giving each target label a certain value in order to be compared and further be classified. These target labels, when grouped, are called clusters. This is what has been done in the previous works in this field. In our project, we performed topic modeling using the technique called LDA or latent dirichlet allocation. LDA assumes each work we are populating into the cluster, is either previously present in some similar chunk or has a new unique meaning so that it can start a new chunk or label on its own. For future improvement or advancements, a new technique called guided lda is under consideration, which is still being worked on.

Related links/reference:-

https://towardsdatascience.com/nlp-extracting-the-main-topics-from-your-dataset-using-lda-in-minutes-21486f5aa925

https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

Emotional Analysis:-

Emotional Analysis in the sense, identifying the core sentiment of the target data. The sentiment can be of three types basically, positive, negative and neutral. When coming to the core emotion driving the sentiment of the target data, there are 12 major types of the emotions which humans express. Unfortunately, since the machines can't identify emotions on their own, we take the help of Natural Language Processing to overcome this limitation. IT basically classifies each and every token generated during the preprocessing of the data, and classifies each token as under which of the identified emotion it comes under. The 12 emotions are neutral, worry, happiness, sadness, love, surprise, fun, relief, hate, empty, enthusiasm, boredom and anger. These when identified and classified upon, provide much deeper insights and scope to further understanding of the data.

Related links/reference:-

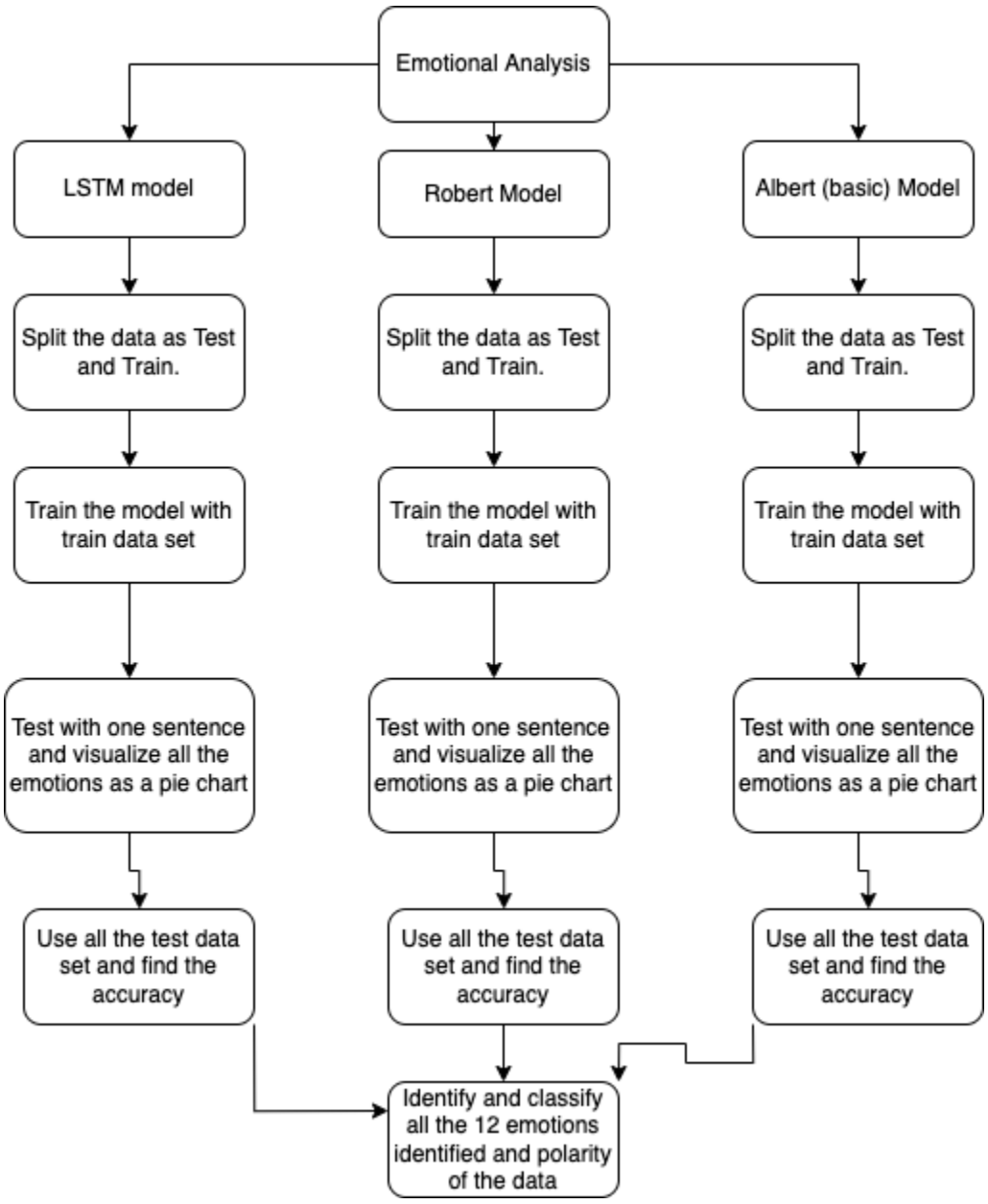
https://www.kaggle.com/c/sa-emotions

Architecture/Workflow model:-

For Million news headlines dataset model:-


Start
Data gathering
Data cleaning
Stop words removal
Conversion of contractions to meaningfull words
Removal of punctuation, special characters
Removal of stop words
Feature Engineering
Visualisation of the features extracted
Text Classification
Zero-shot text based
Sentiment Analysis
Emotional Analysis
Bag of Words
Topic modeling and clustering
Tf-IDF
Model Evaluation and Testing
Stop

For Emotional Analysis:-



Dataset:

1. Million News Headlines dataset:

The dataset we are using is a million news headlines dataset, which contains data of news headlines published over a period of eighteen years. The source of the data is from a reputed Australian news source ABC (Australian Broadcasting Corporation). It is only a single CSV file that contains only two columns, the published_date, and the headline_text.

The dataset contains only two features:

1. publish_date: is the date on which the article is published and the date is in the format yyyyMMdd.

The start and end dates in the dataset are 2003-02-19, 2020-12-31 respectively.

2. headline_text: is a text which contains the headlines of the news published over a period of 18 years.

From the two features, more features are extracted by performing feature engineering:

1. From published_date we can extract the year, month, day respectively as it is in yyyMMdd format.
2. From headline_text we can number of characters, words, the average number of words, count of stopwords, and punctuations present in each headline text.Those are defined as word_count, char_count, mean_word_length, punctuation_count, and stop_word_count.

The below is a representation of the first five rows of the dataset.

	publish_date	headline_text	year	month	date	word_count	char_count	mean_word_length	punctuation_count	stop_word_count
0	20030219	aba decides against community broadcasting lic...	2003	2	19	50	50	7.500000	0	1
1	20030219	act fire witnesses must be aware of defamation	2003	2	19	46	46	4.875000	0	2
2	20030219	a g calls for infrastructure protection summit	2003	2	19	46	46	5.714286	0	2
3	20030219	air nz staff in aust strike for pay rise	2003	2	19	40	40	3.555556	0	2
4	20030219	air nz strike to affect australian travellers	2003	2	19	45	45	5.571429	0	1

2. Emotional Sentiment Analysis:-

For Emotional Sentiment Analysis:- We used another dataset which contains Twitter headlines.

The data set contains 4 features:-

1)Tweet Id:- An ID that contains unique number.

2)Sentiment:- Classified between 12 emotions --> neutral, worry, happiness, sadness, love, surprise, fun, relief, hate, empty, enthusiasm, boredom and anger.

3)Author:-The author who has given the tweet.

4)Content:- It contains the content of the tweet.

	A	B	C	D
	tweet_id	sentiment	author	content
	1956967341	empty	xoshayzers	@tiffanylue i know i was listenin to bad habit earlier and i started freakin at his part =[
	1956967666	sadness	wannamama	Layin n bed with a headache ughhhh...waitin on your call...
	1956967696	sadness	coolfunky	Funeral ceremony...gloomy friday...
	1956967789	enthusiasm	czareaquino	wants to hang out with friends SOON!
	1956968416	neutral	xkilljoyx	@dannycastillo We want to trade with someone who has Houston tickets, but no one will.
	1956968477	worry	xxxPEACHESxxx	Re-pinging @ghostridah14: why didn't you go to prom? BC my bf didn't like my friends
	1956968487	sadness	ShansBee	I should be sleep, but im not! thinking about an old friend who I want. but he's married now. damn, & he wants me 2! scandalous!
	1956968636	worry	mcsleazy	Hmmm. http://www.djhero.com/ is down
	1956969035	sadness	nic0lepaula	@charviray Charlene my love. I miss you
	1956969172	sadness	Ingenue_Em	@kelcouch I'm sorry at least it's Friday?
	1956969456	neutral	feinyheiny	cant fall asleep

Analysis of data:

Pre-processing of data:

Million News Headlines data set:

To analyze the million news headlines dataset, we first read the data, followed by pre-processing of data to know the shape of the dataset, datatypes of each feature and cleaning the dataset if any nulls are present in the dataset.

	publish_date	headline_text
0	20030219	aba decides against community broadcasting lic...
1	20030219	act fire witnesses must be aware of defamation
2	20030219	a g calls for infrastructure protection summit
3	20030219	air nz staff in aust strike for pay rise
4	20030219	air nz strike to affect australian travellers

```
data.shape
```

```
(1226258, 2)
```

```
data.dtypes
```

```
publish_date      int64
headline_text    object
dtype: object
```

```
[5] data.isnull().sum()
```

```
publish_date     0
headline_text    0
dtype: int64
```

As the published_date feature is in the format of yyyyMMdd, we can perform feature engineering to extract the year, month, day respectively. Also, feature engineering can be performed on the headline_text feature, to get the number of characters, words, the average number of words, count of stopwords, and punctuations present in each headline text.

	publish_date	headline_text	year	month	date	word_count	char_count	mean_word_length	punctuation_count	stop_word_count
0	20030219	aba decides against community broadcasting lic...	2003	2	19	50	50	7.500000	0	1
1	20030219	act fire witnesses must be aware of defamation	2003	2	19	46	46	4.875000	0	2
2	20030219	a g calls for infrastructure protection summit	2003	2	19	46	46	5.714286	0	2
3	20030219	air nz staff in aust strike for pay rise	2003	2	19	40	40	3.555556	0	2
4	20030219	air nz strike to affect australian travellers	2003	2	19	45	45	5.571429	0	1

All these features extracted from the existing features can be visualized for a better understanding of their trends and interpretations can be made from those observations. These observations are helpful in performing further analysis of the data. A detailed interpretation of the visualizations is explained in the preliminary results.

For Emotional Analysis:-

For doing the analysis we have to clean the data. For the we have followed the below steps:-

1) Remove incorrect spelling . Ex:- ‘acord’ to ‘accord’

Code:-

```
#using this function we are removing the misspelled words ex:- 'acord': 'accord' (few are listed above)
def misspelled_correction(val):
    for x in val.split():
        if x in miss_corr.keys():
            val = val.replace(x, miss_corr[x])
    return val
```

```
[ ] #cleaning the data i.e making corrections to the mispelled data
    data1["clean_content"] = data1.content.apply(lambda x : misspelled_correction(x))
```

2) Make contractions to meaning full words like ‘I’’ll be’ to ‘I will be’

Code:-

```
#logic for contractions i.e replacing the contractions with its relavent meaning ex:- I'll be --> I
contractions = pd.read_csv("contractions.csv")
index=0
limit = 10000
con =[]
for i in contractions.Contraction :
  for word in i.split():
    con.append(word)

  index += 1
  if index == limit:
    break

for word in con:
  con1 = contractions

cont_dic = dict(zip(contractions.Contraction, contractions.Meaning))
```

```
[ ] #function which does it
    def cont_to_meaning(val):

        for x in val.split():
            if x in cont_dic.keys():
                val = val.replace(x, cont_dic[x])
        return val
```

It is done by splitting each word from the sentence

3) Removing the emojis

Code:-

```
[ ] #removing the emojis in the data
    import emoji
    data1.clean_content = data1.clean_content.apply(lambda x : ' '.join(punctuation(emoji.demojize(x)).split()))
```

```
[ ] #this function cleans(removing the misspelles, converting contractions) the words/characters in a sentence  and again joining the vales/keys
    #to form a meaningfull sentecnce
    def clean_text(val):
        val = misspelled_correction(val)
        val = cont_to_meaning(val)
        val = p.clean(val)
        val = ' '.join(punctuation(emoji.demojize(val)).split())
```

4) Remove URLS and extra spaces:-

Code:-

```
#Removing some important charcters in the data like URLS and empty spaces etc.
p.set_options(p.OPT.MENTION, p.OPT.URL)
```

```
data1["clean_content"]=data1.content.apply(lambda x : p.clean(x))
```

```
#Remove empty comments
data1 = data1[data1.clean_content != ""]
```

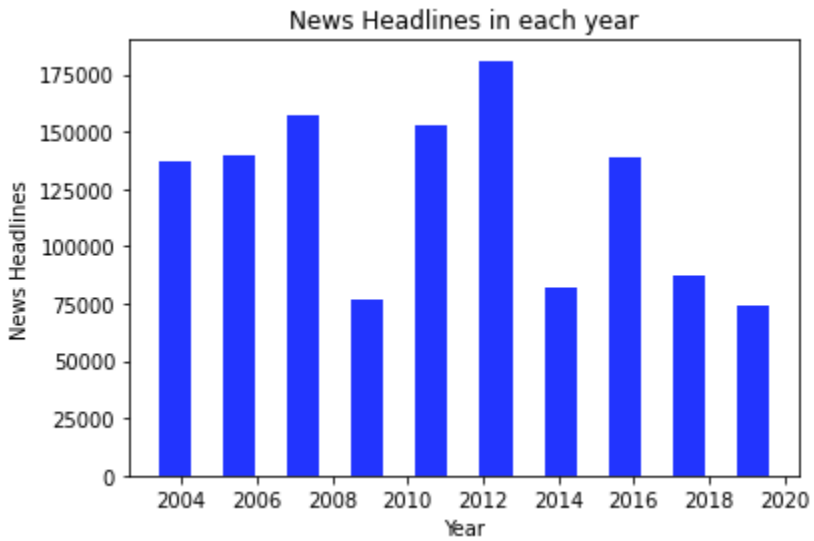
Graph model with explanation:

1.Million News Headlines Dataset:

For a better understanding of the data, the following visualizations are performed.

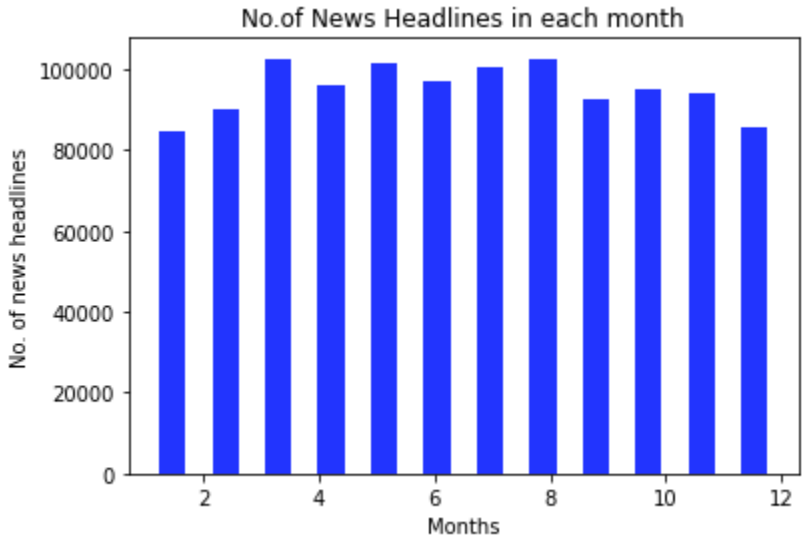
To understand how many news headlines are generated each year, month, and every day, feature engineering is performed on the publish_date feature to extract the date, month, and year columns separately.

Visualizing the news headlines generated each year:



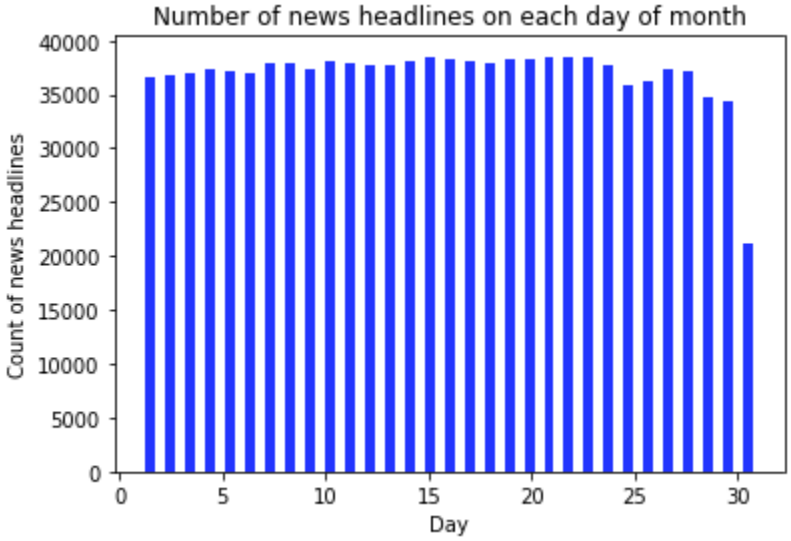
Observation: From the above bar graph, it is understood that the highest news headlines generated was in year 2012 with approximately 175000 news headlines, whereas the lowest generated news headlines was during year 2019 with approximately 60000 news headlines.

Visualizing the news headlines generated each month:



Observation: By observing the above bar graph, we can say that the headlines generated every month varies between 80000 to 100000 headlines.

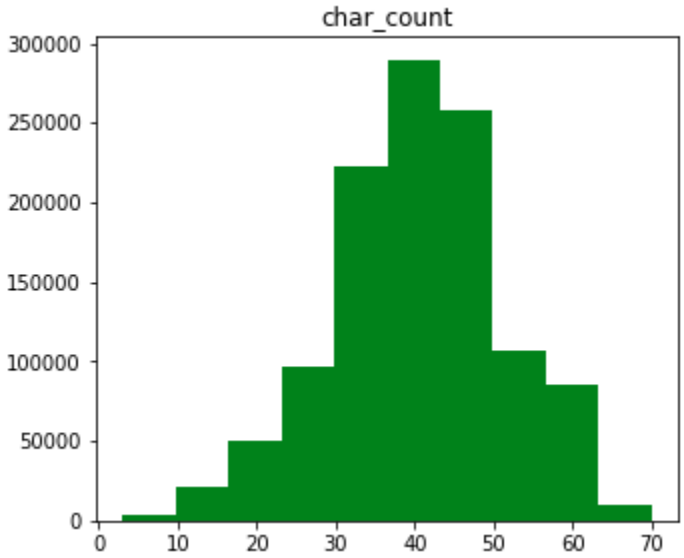
Visualizing the news headlines generated on each day:



Observation: By observing the graph it is understood that there is a uniform distribution in news headlines generated on each day.

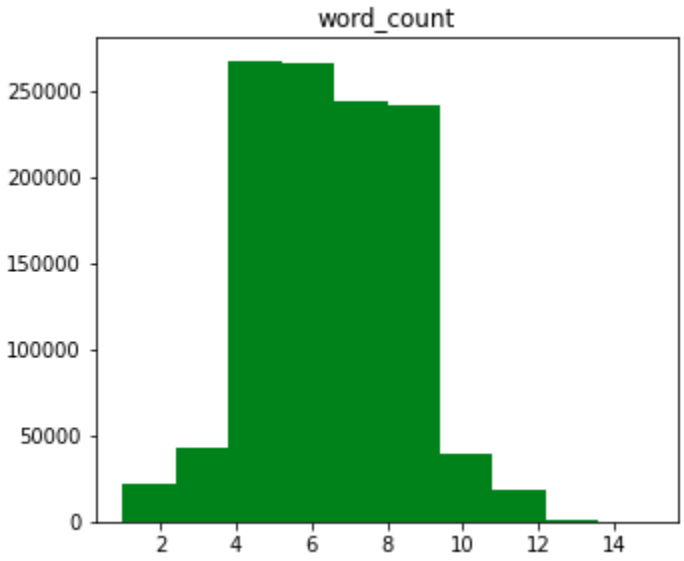
Another step of feature engineering is performed on the headline_text to analyze the distribution of the number of words, characters, punctuations present in the headline text. Moreover, a mean length of the words in the headline text and distribution of the count of stop words in the headline_text is also visualized.

Distribution of the number of characters in the headline text:



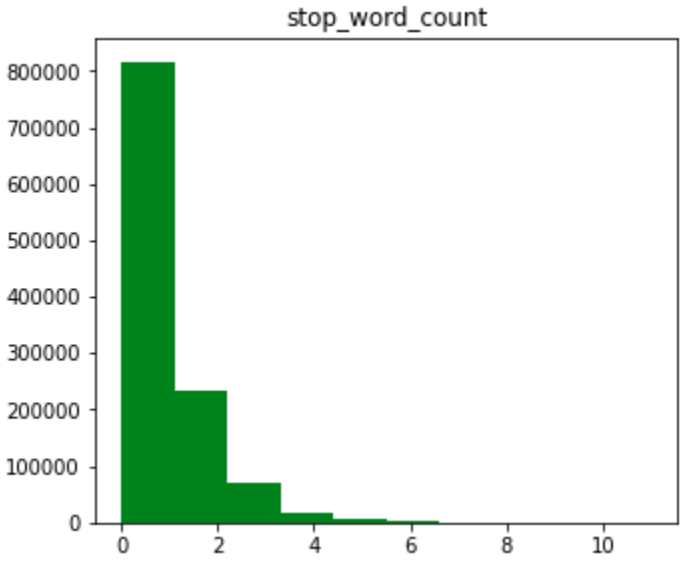
Observation: From the histogram, we can interpret that in general, the range of characters is from 2 to 70, however, the maximum number of characters present in the headline text lies between 30 to 50 approximately.

Distribution of the number of words present in the headline text:



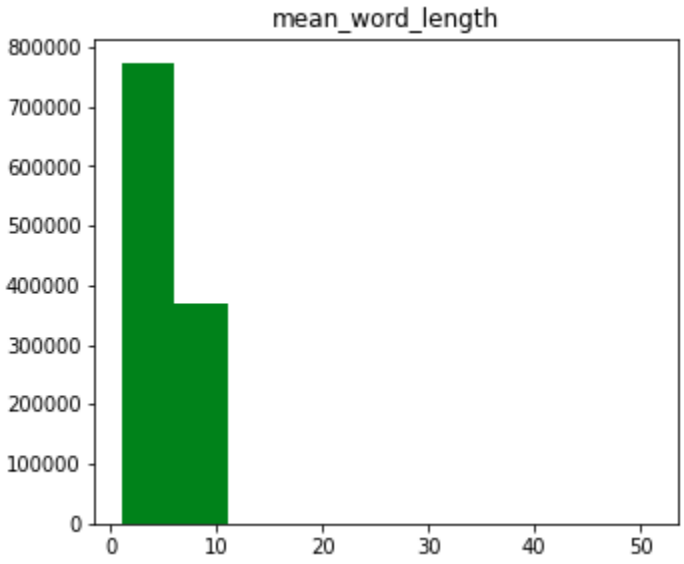
Observation: From the distribution of histogram it is observed that the range of words present in headline text is from 2 to 12, whereas the maximum number of words present in the headline text is between 4 and 9 approximately with a count range between 220000 to 270000.

Distribution of the number of stopwords in the headline text:



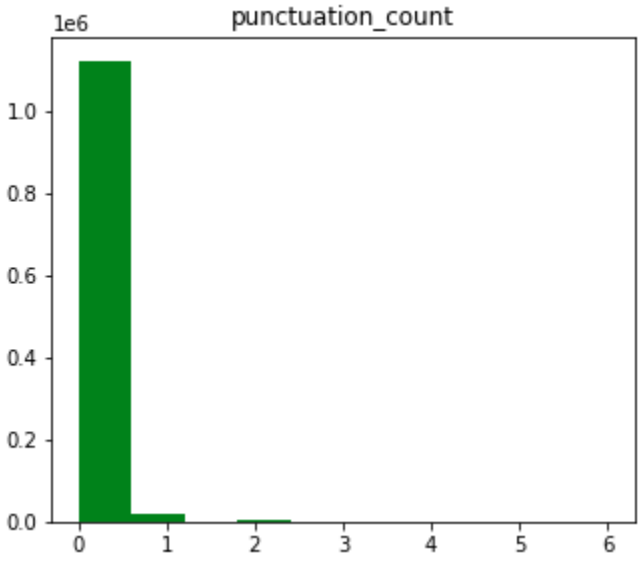
Observation: the stopwords range is from 0 to 6 in general, the maximum number of stop words are between 0 to 2 with a maximum count of 800000.

Distribution of average word length in the headline text:



Observation: From the histogram it is observed that the average length of the word ranges between 1to 10 with a maximum word length ranging from 1 to 5 with a count of 750000.

Distribution of the number of punctuations present in the headline text:

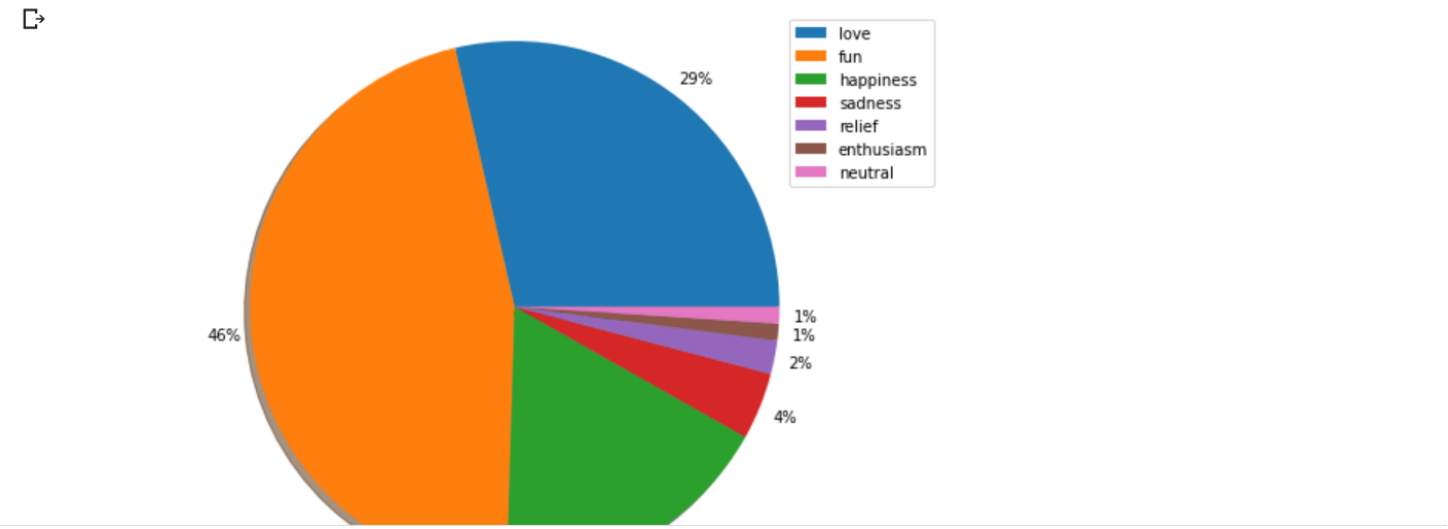


Observation: The histogram distribution of punctuation count in exponential form interprets that, the punctuation marks in the headline text range between 0 to 2, where the maximum range of punctuations in a headline is between 0 to 8.

2. Emotional Analysis for twitter dataset:-

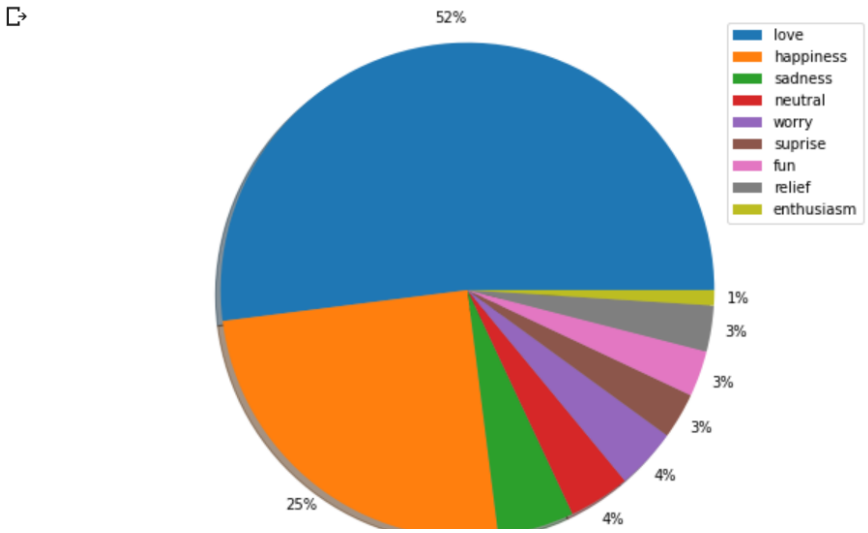
Emotional analysis of a sentence using LSTM model.

```
#testing the model with a given sentence
result =get_sentiment(model,"Had an absolutely brilliant day ðŸ˜ loved seeing an old friend and reminiscing")
plot_result(result)
```



Emotional analysis of a sentence using Robert model.

```
#testing the model with a given sentence same as used above
result =get_sentiment(model_lstm_gwe,"Had an absolutely brilliant day ðŸ˜ loved seeing an old friend and reminiscing")
plot_result(result)
```



Implementation:

1.Million News Headlines dataset:

After reading, pre-processing, extracting features by performing feature engineering. We performed visualizations for a better understanding of the data and to interpret the observations.

Text Classification:

The next step performed is text classification and to detect sentiment in the text in an unsupervised way. We choose Zero-shot text classification with Hugging face. Zero-shot model on some predefined topics without training data but helps in building our model.

```
classifier = pipeline("zero-shot-classification", device=0)
```

To build our model we decided to limit the length of our model to only 1000 as
the dataset is huge and contains 1226258 articles, which consumes more time. The 1000 articles sample is defined as **`sample_headline_text`** and we choose five labels are renewable, politics, emission, temperature, emergency, advertisement.

```
sample_headline_text=df['headline_text'][:1000]
candidate_labels = ["renewable", "politics", "emission", "temperature", "emergency",
 "advertisment"]
```

After defining the labels we can run the classifier to assign a probability to each label. We are using a default option where the pipeline assumes only one candidate label is true by returning a list of scores for each label that upon addition becomes 1.
A loop is written to iterate over the 1000 articles and we get an output as a list of probabilities for each label for every article. If the score for classification is greater than 0.5 it will be moved to the next label.

```
for sent in sample_headline_text.values:
    res = classifier(sent, candidate_labels)
```

As the output we receive is the classification score for the five labels defined, we can covert the candidate labels and candidate results as a dictionary and then convert it as a data frame which helps in visualizing the count for each label for all the 1000 articles we choose as samples. The below code helps us achieve that.

```
labels_scores = {'labels': candidate_labels,
        'values': candidate_results}

df_chart = pd.DataFrame(labels_scores, columns=['labels','values'])

df_chart
```

The bar graph is also plotted which will be discussed in the results section.

Sentiment Analysis:

To perform sentiment analysis over the headline_text, we first created a corpus of text and then we performed tokenization by importing nltk, and then applying word_tokenize on the corpus.

```
corpus = str()
for i in range(len(df['headline_text'])):
    corpus += (' ')+df['headline_text'][i]
# Toenization
import nltk
nltk.download('punkt')
words = nltk.word_tokenize(corpus)
```

Since the data contains punctuations and stopwords which are not really important for sentiment analysis, so we cleaned the data by removing the punctuations and stopwords.

```
# removing stopwords and punctuations
f_words = [w for w in words if not w in english_stops]

punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
fp_words = [w for w in f_words if not w in punctuations]
```

The frequency distribution of the clean words is done and the top 15 words are visualized and will be discussed in the results section.

The sample_headline_text of 1000 samples which we used for text classification is used for performing the sentiment analysis. The sample is converted into a list and is defined as a sequence. The Zero-shot hugging face classifier which was previously used for text classification is only used here for sentiment analysis by defining sentiment_labels a list with positive and negative and passing the sequence and the list of sentiment_labels to the classifier.
Below is the code :

```
sentiment_labels = ['positive','negative']
sequence = list(sample_headline_text.values)
sent = classifier(sequence, sentiment_labels)
```

The output when sent is printed is a sequence which is an article, labels: positive, negative and the scores for their respective scores. A score above 0.5 is negative and less than 0.5 is positive. For every article, both positive and negative scores are returned, by looking at the score we can say which article is a negative article or positive.

Topic Modelling and Clustering:

Our aim for performing topic model and clustering analysis is to identify the topics in the dataset and then make predictions for the unseen data.

Preprocessing of text and clustering:

As a part of preprocessing and text clustering, 10000 random records from the data set are taken since the dataset is huge.

```
data = df.sample(10000, random_state=42)
```

To extract root words stemming is performed on records using Snowball stemmer. Preprocessing which involves cleaning of text is performed by using the genism library and stop words are eliminated from text.
The following code helps to turn the text into bag-of-words representation and then techniques like lemmatization and stemming are performed to get each word into a common base or root word.

```
import gensim
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS
from nltk.stem import WordNetLemmatizer, SnowballStemmer
import numpy as np
np.random.seed(42)


def lemmatize_stemming(doc):
    stemmer = SnowballStemmer('english')
    return stemmer.stem(WordNetLemmatizer().lemmatize(doc, pos='v'))

def text_preprocess(doc):
    # min_len and max_len: minimum and maximum lengths of token (inclusive)
    processed_words = gensim.utils.simple_preprocess(doc,min_len=3, max_len=15)
    stop_words = gensim.parsing.preprocessing.STOPWORDS
    tokens = [lemmatize_stemming(token) for token in processed_words if token not in stop_wo

    return tokens
```

Text processing is applied on the headline_text which is defined as processed_docs. This is basically creating a corpus.

We then convert the corpus into vectors, we can perform this in two ways.

1. The bag-of-words representation uses word count and the order of the word does not matter.

2. TF-IDF transforms the word count of bag-of-words into the weight of the word.

1. Bag of Words:
A dictionary is created by mapping the processed words and the integer id's and then filtering those based on few parameters.
Corpus that is extracted by removing stop words is converted into vectors using Bag-of-words. Token ids along with their counts are printed using the doc2bow technique.

```
# create a dictionary - a mapping between words and their integer ids

dictionary = gensim.corpora.Dictionary(processed_docs)
dictionary.filter_extremes(no_below=15, no_above=0.6, keep_n=5000)
#doc2bow converts document into the bag-of-words (BoW) format, which is a list of
 (token_id, token_count) tuples.

bow_corpus = [dictionary.doc2bow(doc) for doc in processed_docs]
Bow_corpus[8]
```

The above code results in tuples with token ids and the number of times they appeared in document 8.

2. TF-IDF
To convert words into vectors based on their weights, TFIDF vectorization is implemented by importing models in the genism library. Likewise in Bag-of-words, ids words and vector weights from TFIDF vectorization are printed.

Topic Modelling:
We implement topic modeling using LdaMulticore from genism library for bow_corpus resulted from bag-of-words and corpus_tfidf resulted from TFIDF vectorization techniques and divide all records in data set into 20 topics and print topics and vectors for words in each topic.

Model Evaluation:
To analyze the performance of model that is implemented on bow_corpus and corpus_tfidf model evaluation is performed in which scores are printed for topics in sorted order. From which it can be concluded that TFIDF performs similar to Bag-of-Words.

Model Testing:
Model testing is performed on documents which are not present in training data. Data is tested using doc2bow and ids and counts are printed. Lda_model is implemented on testing data and data is classified into topics and conclusions are made on sentences based on topic they result.

2. Emotional Analysis for twitter dataset:-

Algorithm and Explanation of Implementation:-
Basically we are using the twitter data to find the emotions. We have 12 emotions included in this they are (neutral, worry, happiness, sadness, love, surprise, fun, relief, hate, empty, enthusiasm, boredom and anger)

The Flow of the algorithm can be seen in the architecture/work flow diagram above.

We have taken a data set has explained above and made the data clean. That includes removing emojis, punctuation,spaces, Characters.

After we have splitted the data as training and test set and trained the LSTM and Robert model.

After training the model we have tested few sentences for each model and seen what emotions are predicted from it. The Images are shown in before points.

After that we have found the accuracy of the all 2 models

For testing purpose we have also tried Albert model and found its accuracy.

Code for training the model for LSTM:-

```
#training the model
model_lstm_gwe.fit(X_train_pad, y_train, epochs = Epoch, batch_size=batch_size,validation_data=(X_test_pad, y_test))
```

```
Epoch 1/5
1000/1000 [==============================] - 1136s 1s/step - loss: 2.1472 - accuracy: 0.2342 - val_loss: 2.1079 - val_accuracy: 0.2510
Epoch 2/5
1000/1000 [==============================] - 1131s 1s/step - loss: 2.1056 - accuracy: 0.2593 - val_loss: 2.0935 - val_accuracy: 0.2537
Epoch 3/5
1000/1000 [==============================] - 1131s 1s/step - loss: 2.0790 - accuracy: 0.2694 - val_loss: 2.0920 - val_accuracy: 0.2626
Epoch 4/5
1000/1000 [==============================] - 1129s 1s/step - loss: 2.0529 - accuracy: 0.2791 - val_loss: 2.0818 - val_accuracy: 0.2648
Epoch 5/5
1000/1000 [==============================] - 1130s 1s/step - loss: 2.0233 - accuracy: 0.2903 - val_loss: 2.0754 - val_accuracy: 0.2720
<keras.callbacks.History at 0x7fb80596c250>
```

Code for training the model for Robert:-

```
#training the model
model.fit(X_train_pad, y_train, epochs = Epoch, batch_size=batch_size,validation_data=(X_test_pad, y_test))
```

```
Epoch 1/5
1000/1000 [==============================] - 1324s 1s/step - loss: 2.1313 - accuracy: 0.2443 - val_loss: 2.0981 - val_accuracy: 0.2567
Epoch 2/5
1000/1000 [==============================] - 1316s 1s/step - loss: 1.9911 - accuracy: 0.3138 - val_loss: 2.0757 - val_accuracy: 0.2754
Epoch 3/5
1000/1000 [==============================] - 1318s 1s/step - loss: 1.6751 - accuracy: 0.4377 - val_loss: 2.2286 - val_accuracy: 0.2522
Epoch 4/5
1000/1000 [==============================] - 1315s 1s/step - loss: 1.3273 - accuracy: 0.5523 - val_loss: 2.4589 - val_accuracy: 0.2467
Epoch 5/5
1000/1000 [==============================] - 1311s 1s/step - loss: 1.0753 - accuracy: 0.6328 - val_loss: 2.7095 - val_accuracy: 0.2304
<keras.callbacks.History at 0x7fb80c164b10>
```

Remaining of code is in the GITHUB as
abc_news_final_code.ipynb for Million News headlines dataset.
twitter_emotion_sentiment-analyis.ipynb for Twitter dataset.

Results:

1. Million News headlines dataset:

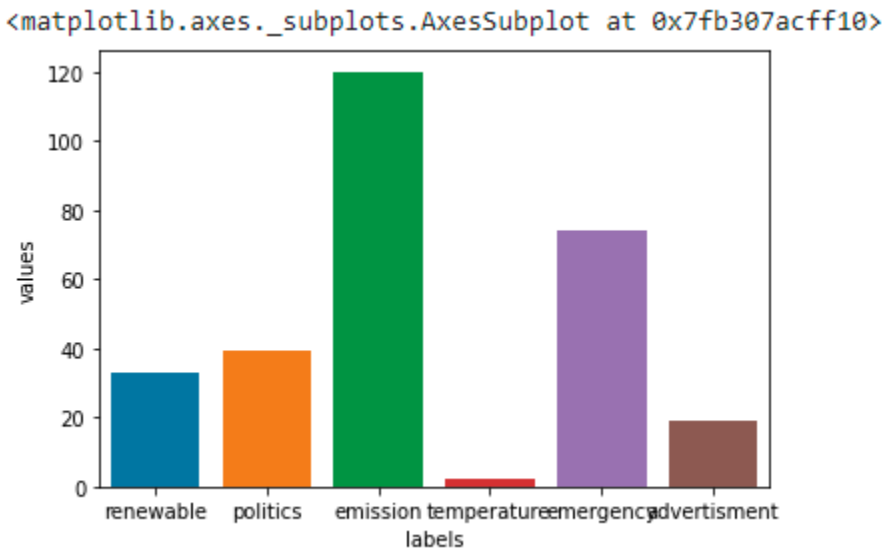
Text Classification:

Below is the output for the text classification done by using zero-shot hugging face classification, by pre-defining 5 labels for each headline for the 1000 sample headlines selected.

```
act fire witnesses must be aware of defamation
['emission', 'emergency', 'temperature', 'renewable', 'advertisment', 'politics']
[0.671912670135498, 0.12633615732192993, 0.11947888880968094, 0.06294111162424088, 0.011076904833316803, 0.00825420580804348]

air nz staff in aust strike for pay rise
['emission', 'temperature', 'emergency', 'advertisment', 'politics', 'renewable']
[0.5271357297897339, 0.17796404659748077, 0.14966486394405365, 0.09618441015481949, 0.032544929534196854, 0.016505971550941467]

antic delighted with record breaking barca
['emission', 'advertisment', 'temperature', 'emergency', 'renewable', 'politics']
[0.6439234018325806, 0.1459626853466034, 0.10467113554477692, 0.05406339094042778, 0.040799692273139954, 0.010579731315374374]

/usr/local/lib/python3.7/dist-packages/transformers/pipelines/base.py:910: UserWarning: You seem to be using the pipelines sequential
  UserWarning,
bushfire victims urged to see centrelink
['emergency', 'emission', 'temperature', 'advertisment', 'renewable', 'politics']
[0.5025433897972107, 0.2497611939907074, 0.10255458205938339, 0.09804937243461609, 0.02919478341937065, 0.017896637320518494]

call for ethanol blend fuel to go ahead
['renewable', 'advertisment', 'emission', 'emergency', 'politics', 'temperature']
[0.600139319896698, 0.19594863057136536, 0.09688589721918106, 0.04681083932518959, 0.041064392775297165, 0.01915096864104271]

cemeteries miss out on funds
['emission', 'temperature', 'emergency', 'advertisment', 'renewable', 'politics']
[0.6596245169639587, 0.2605804204940796, 0.03498111665248871, 0.027310488745570183, 0.009001380763947964, 0.008502118289470673]
```

The below is the data frame representation for all the labels defined which shows labels on the left and values on right side that shows the values of each predefined label. A barplot is also plotted for visualizing the labels using Sns barplot.

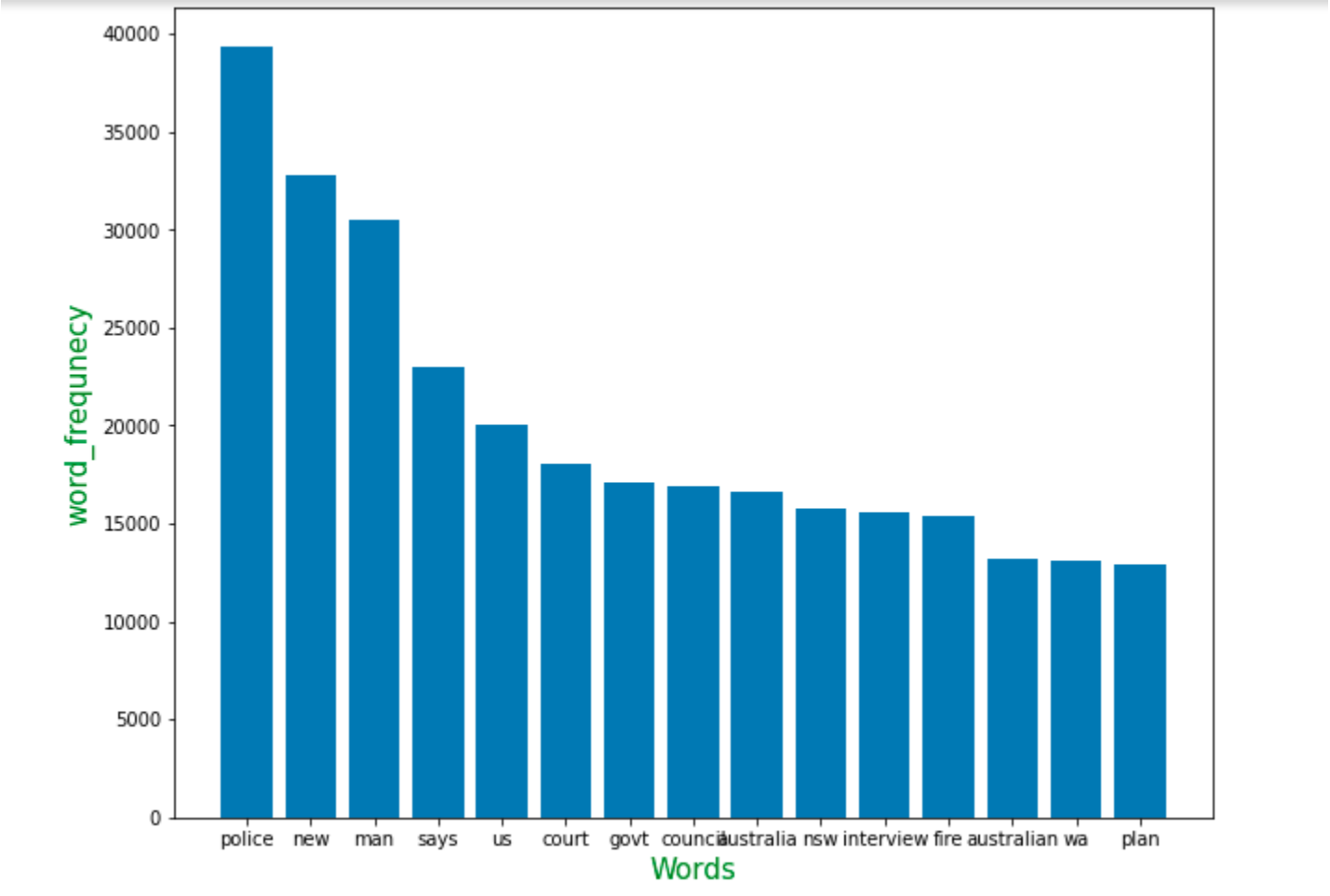
	labels	values
0	renewable	33
1	politics	39
2	emission	120
3	temperature	2
4	emergency	74
5	advertisment	19



From the above observation it is clear that the 1000 sample headlines selected fall in the emission label.

Sentiment Analysis:

The below is the frequency distribution plot for the top 15 words for the clean words.



The below is the result of the sentiment analysis which is performed on the sample headline text of 1000 headlines Which is passed to the zero-shot hugging face classifier with the sequence and labels. The output here is a List with a dictionary for each sequence along with the labels: negative and positive and also their scores.
If the classifier score is greater than 0.5 it is negative sequence and less than 0.5 score is a positive sequence.


```
[{'labels': ['negative', 'positive'],
  'scores': [0.9875787496566772, 0.012421276420354843],
  'sequence': 'aba decides against community broadcasting licence'},
 {'labels': ['negative', 'positive'],
  'scores': [0.99607253074646, 0.003927456680685282],
  'sequence': 'act fire witnesses must be aware of defamation'},
 {'labels': ['negative', 'positive'],
  'scores': [0.6098466515541077, 0.39015331864356995],
  'sequence': 'a g calls for infrastructure protection summit'},
 {'labels': ['negative', 'positive'],
  'scores': [0.9354528784751892, 0.0645470842719078],
  'sequence': 'air nz staff in aust strike for pay rise'},
 {'labels': ['negative', 'positive'],
  'scores': [0.9901187419891357, 0.00988126639276743],
  'sequence': 'air nz strike to affect australian travellers'},
 {'labels': ['positive', 'negative'],
  'scores': [0.9616013765335083, 0.03839866444468498],
  'sequence': 'ambitious olsson wins triple jump'},
 {'labels': ['positive', 'negative'],
  'scores': [0.9938040971755981, 0.006195914000272751],
  'sequence': 'antic delighted with record breaking barca'},
```


Topic Modeling with LDA:

1.Bag-of Words :

The below is the output of topic modelling with LDA using bag-of-words corpus, by selecting only 20 topics for the 10,000 headlines randomly selected.
By looking at the output of each topic we can understand what the topic is about.

```
Topic0:
0.126*"new" + 0.068*"hit" + 0.056*"australian" + 0.048*"industri" + 0.033*"high" + 0.030*"safeti" + 0.029*"head" + 0.024*"hope" + 0.024*"challeng" + 0.022*"ener
Topic1:
0.075*"car" + 0.072*"test" + 0.064*"child" + 0.059*"accus" + 0.050*"sex" + 0.041*"guilti" + 0.038*"abus" + 0.035*"children" + 0.032*"men" + 0.031*"tas"
Topic2:
0.054*"coronavirus" + 0.051*"famili" + 0.044*"port" + 0.040*"inquiri" + 0.039*"food" + 0.031*"hear" + 0.030*"make" + 0.029*"darwin" + 0.027*"teen" + 0.027*"gas"
Topic3:
0.130*"plan" + 0.097*"interview" + 0.036*"protest" + 0.034*"union" + 0.032*"welcom" + 0.032*"park" + 0.032*"victim" + 0.029*"bushfir" + 0.028*"beat" + 0.024*"vi
Topic4:
0.089*"australia" + 0.057*"hous" + 0.051*"crash" + 0.048*"continu" + 0.037*"public" + 0.034*"resid" + 0.034*"aussi" + 0.030*"probe" + 0.030*"centr" + 0.023*"res
Topic5:
0.167*"man" + 0.085*"charg" + 0.063*"jail" + 0.062*"year" + 0.060*"attack" + 0.055*"die" + 0.054*"hospit" + 0.046*"murder" + 0.027*"fall" + 0.022*"act"
Topic6:
0.121*"say" + 0.086*"death" + 0.065*"sydney" + 0.064*"day" + 0.041*"price" + 0.036*"look" + 0.036*"flood" + 0.035*"rise" + 0.034*"road" + 0.033*"pay"
Topic7:
0.083*"govt" + 0.067*"urg" + 0.050*"elect" + 0.049*"school" + 0.048*"china" + 0.046*"servic" + 0.044*"health" + 0.038*"defend" + 0.033*"labor" + 0.029*"boost"
Topic8:
0.051*"south" + 0.050*"nation" + 0.048*"concern" + 0.046*"open" + 0.039*"rural" + 0.038*"mine" + 0.036*"support" + 0.036*"leav" + 0.035*"lead" + 0.030*"compani"
Topic9:
0.059*"call" + 0.053*"warn" + 0.040*"push" + 0.039*"deal" + 0.039*"state" + 0.029*"give" + 0.028*"scheme" + 0.026*"life" + 0.026*"race" + 0.026*"ahead"
Topic10:
0.070*"miss" + 0.048*"search" + 0.047*"green" + 0.046*"adelaid" + 0.039*"get" + 0.036*"leader" + 0.035*"hold" + 0.028*"trump" + 0.028*"perth" + 0.027*"game"
Topic11:
0.060*"market" + 0.045*"minist" + 0.044*"talk" + 0.042*"cut" + 0.039*"job" + 0.037*"forc" + 0.036*"farm" + 0.034*"busi" + 0.032*"lose" + 0.032*"storm"
Topic12:
0.067*"govern" + 0.064*"home" + 0.058*"queensland" + 0.052*"melbourn" + 0.051*"shoot" + 0.048*"investig" + 0.047*"world" + 0.047*"cup" + 0.046*"polic" + 0.045*"
```



```
Topic13:
0.113*"kill" + 0.051*"dead" + 0.043*"train" + 0.036*"despit" + 0.034*"senat" + 0.029*"start" + 0.029*"club" + 0.028*"spark" + 0.027*"confid" + 0.027*"expect"
Topic14:
0.087*"win" + 0.082*"nsw" + 0.074*"report" + 0.048*"fear" + 0.044*"set" + 0.042*"final" + 0.034*"feder" + 0.030*"ban" + 0.028*"communiti" + 0.026*"prepar"
Topic15:
0.084*"council" + 0.074*"face" + 0.063*"help" + 0.061*"woman" + 0.056*"farmer" + 0.054*"claim" + 0.036*"end" + 0.033*"need" + 0.031*"reject" + 0.030*"rain"
Topic16:
0.176*"polic" + 0.080*"water" + 0.048*"seek" + 0.035*"offic" + 0.034*"brisban" + 0.034*"fight" + 0.034*"work" + 0.031*"budget" + 0.027*"target" + 0.027*"appeal"
Topic17:
0.046*"case" + 0.043*"return" + 0.041*"trial" + 0.037*"countri" + 0.037*"releas" + 0.035*"prison" + 0.035*"secur" + 0.034*"close" + 0.027*"question" + 0.026*"ho
Topic18:
0.126*"court" + 0.066*"qld" + 0.044*"law" + 0.038*"drug" + 0.037*"big" + 0.036*"arrest" + 0.034*"alleg" + 0.034*"decis" + 0.030*"assault" + 0.025*"tourism"
Topic19:
0.096*"chang" + 0.055*"coast" + 0.053*"war" + 0.039*"east" + 0.037*"gold" + 0.036*"sale" + 0.034*"land" + 0.034*"west" + 0.026*"manag" + 0.025*"educ"
```

2. TF-IDF :

The below is the output of topic modelling with LDA using `corpus_tfidf` corpus, by selecting only 20 topics for the 10,000 headlines randomly selected.

```
Topic0:
0.078*"hit" + 0.060*"search" + 0.042*"train" + 0.042*"prison" + 0.041*"head" + 0.037*"farmer" + 0.035*"life" + 0.031*"fail" + 0.028*"match" + 0.028*"energi"
Topic1:
0.056*"test" + 0.047*"child" + 0.040*"car" + 0.039*"countri" + 0.036*"sex" + 0.034*"servic" + 0.033*"abc" + 0.032*"inquiri" + 0.030*"children" + 0.030*"abus"
Topic2:
0.057*"school" + 0.051*"shoot" + 0.039*"port" + 0.032*"despit" + 0.032*"research" + 0.026*"violenc" + 0.026*"teen" + 0.026*"sell" + 0.025*"week" + 0.024*"miner"
Topic3:
0.157*"interview" + 0.047*"return" + 0.034*"union" + 0.034*"welcom" + 0.032*"victim" + 0.032*"park" + 0.028*"star" + 0.026*"centr" + 0.025*"station" + 0.022*"pl
Topic4:
0.049*"hous" + 0.047*"crash" + 0.043*"aussi" + 0.041*"australia" + 0.034*"public" + 0.032*"continu" + 0.032*"road" + 0.032*"probe" + 0.031*"beat" + 0.024*"make"
Topic5:
0.064*"die" + 0.055*"attack" + 0.041*"protest" + 0.041*"leav" + 0.038*"storm" + 0.036*"man" + 0.036*"famili" + 0.035*"murder" + 0.035*"hospit" + 0.030*"fall"
Topic6:
0.068*"day" + 0.061*"death" + 0.043*"look" + 0.040*"coronavirus" + 0.039*"get" + 0.036*"new" + 0.032*"crime" + 0.032*"mayor" + 0.031*"second" + 0.029*"land"
Topic7:
0.051*"china" + 0.043*"elect" + 0.041*"defend" + 0.040*"state" + 0.037*"take" + 0.036*"deal" + 0.031*"boost" + 0.031*"consid" + 0.029*"speak" + 0.029*"studi"
Topic8:
0.042*"concern" + 0.039*"job" + 0.039*"south" + 0.038*"rural" + 0.033*"support" + 0.032*"feder" + 0.031*"cut" + 0.029*"open" + 0.028*"offer" + 0.025*"go"
Topic9:
0.035*"push" + 0.035*"sydney" + 0.032*"group" + 0.030*"warn" + 0.030*"tax" + 0.029*"strike" + 0.026*"race" + 0.026*"reject" + 0.025*"hear" + 0.024*"scheme"
Topic10:
0.064*"miss" + 0.050*"accus" + 0.034*"adelaid" + 0.034*"hold" + 0.033*"tell" + 0.032*"time" + 0.032*"share" + 0.029*"rule" + 0.028*"north" + 0.027*"prepar"
Topic11:
0.053*"market" + 0.048*"nation" + 0.047*"minist" + 0.044*"flood" + 0.041*"news" + 0.040*"forc" + 0.039*"rat" + 0.035*"busi" + 0.030*"power" + 0.026*"season"
Topic12:
0.042*"home" + 0.041*"world" + 0.040*"lead" + 0.039*"cup" + 0.035*"melbourn" + 0.034*"queensland" + 0.030*"guilti" + 0.029*"student" + 0.027*"back" + 0.026*"dri
```

```
Topic13:
0.085*"australian" + 0.062*"dead" + 0.038*"deni" + 0.031*"senat" + 0.031*"tiger" + 0.030*"parti" + 0.030*"start" + 0.030*"titl" + 0.028*"expect" + 0.027*"club"
Topic14:
0.058*"win" + 0.055*"water" + 0.043*"nsw" + 0.042*"set" + 0.041*"report" + 0.038*"final" + 0.037*"leader" + 0.036*"fear" + 0.032*"review" + 0.031*"lose"
Topic15:
0.057*"charg" + 0.054*"court" + 0.045*"help" + 0.041*"woman" + 0.037*"case" + 0.035*"claim" + 0.033*"man" + 0.030*"seek" + 0.028*"drug" + 0.028*"need"
Topic16:
0.051*"face" + 0.043*"investig" + 0.043*"fight" + 0.041*"work" + 0.037*"brisban" + 0.036*"polic" + 0.032*"opposit" + 0.029*"appeal" + 0.029*"rescu" + 0.027*"riv
Topic17:
0.051*"talk" + 0.044*"trial" + 0.042*"farm" + 0.039*"releas" + 0.039*"food" + 0.038*"close" + 0.037*"end" + 0.035*"secur" + 0.032*"labor" + 0.030*"premier"
Topic18:
0.050*"jail" + 0.049*"qld" + 0.036*"law" + 0.034*"urg" + 0.034*"resid" + 0.033*"big" + 0.033*"assault" + 0.032*"offic" + 0.031*"arrest" + 0.031*"decis"
Topic19:
0.059*"chang" + 0.044*"green" + 0.042*"meet" + 0.040*"coast" + 0.033*"sale" + 0.033*"call" + 0.028*"war" + 0.027*"stand" + 0.027*"hope" + 0.027*"victoria"
```

Model Evaluation:

1. Bag-of-words:

By the applying the bag-of-words to the lda model the following is the resulting output.
If we evaluate our model with the previous obtained outputs most of them match.
For example:
Topic:4 it contains words like australia, public, research, crash etc., all of them are same.

```
7, 0.6100000143051147, 0.083*"govt" + 0.067*"urg" + 0.050*"elect" + 0.049*"school" + 0.048*"china"
9, 0.21000000834465027, 0.059*"call" + 0.053*"warn" + 0.040*"push" + 0.039*"deal" + 0.039*"state"
0, 0.010000000707805157, 0.126*"new" + 0.068*"hit" + 0.056*"australian" + 0.048*"industri" + 0.033*"high"
1, 0.010000000707805157, 0.075*"car" + 0.072*"test" + 0.064*"child" + 0.059*"accus" + 0.050*"sex"
2, 0.010000000707805157, 0.054*"coronavirus" + 0.051*"famili" + 0.044*"port" + 0.040*"inquiri" + 0.039*"food"
3, 0.010000000707805157, 0.130*"plan" + 0.097*"interview" + 0.036*"protest" + 0.034*"union" + 0.032*"welcom"
4, 0.010000000707805157, 0.089*"australia" + 0.057*"hous" + 0.051*"crash" + 0.048*"continu" + 0.037*"public"
5, 0.010000000707805157, 0.167*"man" + 0.085*"charg" + 0.063*"jail" + 0.062*"year" + 0.060*"attack"
6, 0.010000000707805157, 0.121*"say" + 0.086*"death" + 0.065*"sydney" + 0.064*"day" + 0.041*"price"
8, 0.010000000707805157, 0.051*"south" + 0.050*"nation" + 0.048*"concern" + 0.046*"open" + 0.039*"rural"
10, 0.010000000707805157, 0.070*"miss" + 0.048*"search" + 0.047*"green" + 0.046*"adelaid" + 0.039*"get"
11, 0.010000000707805157, 0.060*"market" + 0.045*"minist" + 0.044*"talk" + 0.042*"cut" + 0.039*"job"
12, 0.010000000707805157, 0.067*"govern" + 0.064*"home" + 0.058*"queensland" + 0.052*"melbourn" + 0.051*"shoot"
13, 0.010000000707805157, 0.113*"kill" + 0.051*"dead" + 0.043*"train" + 0.036*"despit" + 0.034*"senat"
14, 0.010000000707805157, 0.087*"win" + 0.082*"nsw" + 0.074*"report" + 0.048*"fear" + 0.044*"set"
15, 0.010000000707805157, 0.084*"council" + 0.074*"face" + 0.063*"help" + 0.061*"woman" + 0.056*"farmer"
16, 0.010000000707805157, 0.176*"polic" + 0.080*"water" + 0.048*"seek" + 0.035*"offic" + 0.034*"brisban"
17, 0.010000000707805157, 0.046*"case" + 0.043*"return" + 0.041*"trial" + 0.037*"countri" + 0.037*"releas"
18, 0.010000000707805157, 0.126*"court" + 0.066*"qld" + 0.044*"law" + 0.038*"drug" + 0.037*"big"
19, 0.010000000707805157, 0.096*"chang" + 0.055*"coast" + 0.053*"war" + 0.039*"east" + 0.037*"gold"
```

2. TF-IDF:

By the applying the corpus_tfidf to the lda model the following is the resulting output.
If we evaluate our model with the previous obtained outputs most of them match.
For example:
Topic:4 it contains words like australia, public, research, crash etc., all of them are same.

```
7, 0.6825274229049683, 0.083*"govt" + 0.067*"urg" + 0.050*"elect" + 0.049*"school" + 0.048*"china"
0, 0.016709083691239357, 0.126*"new" + 0.068*"hit" + 0.056*"australian" + 0.048*"industri" + 0.033*"high"
1, 0.016709083691239357, 0.075*"car" + 0.072*"test" + 0.064*"child" + 0.059*"accus" + 0.050*"sex"
2, 0.016709083691239357, 0.054*"coronavirus" + 0.051*"famili" + 0.044*"port" + 0.040*"inquiri" + 0.039*"food"
3, 0.016709083691239357, 0.130*"plan" + 0.097*"interview" + 0.036*"protest" + 0.034*"union" + 0.032*"welcom"
4, 0.016709083691239357, 0.089*"australia" + 0.057*"hous" + 0.051*"crash" + 0.048*"continu" + 0.037*"public"
5, 0.016709083691239357, 0.167*"man" + 0.085*"charg" + 0.063*"jail" + 0.062*"year" + 0.060*"attack"
6, 0.016709083691239357, 0.121*"say" + 0.086*"death" + 0.065*"sydney" + 0.064*"day" + 0.041*"price"
8, 0.016709083691239357, 0.051*"south" + 0.050*"nation" + 0.048*"concern" + 0.046*"open" + 0.039*"rural"
9, 0.016709083691239357, 0.059*"call" + 0.053*"warn" + 0.040*"push" + 0.039*"deal" + 0.039*"state"
10, 0.016709083691239357, 0.070*"miss" + 0.048*"search" + 0.047*"green" + 0.046*"adelaid" + 0.039*"get"
11, 0.016709083691239357, 0.060*"market" + 0.045*"minist" + 0.044*"talk" + 0.042*"cut" + 0.039*"job"
12, 0.016709083691239357, 0.067*"govern" + 0.064*"home" + 0.058*"queensland" + 0.052*"melbourn" + 0.051*"shoot"
13, 0.016709083691239357, 0.113*"kill" + 0.051*"dead" + 0.043*"train" + 0.036*"despit" + 0.034*"senat"
14, 0.016709083691239357, 0.087*"win" + 0.082*"nsw" + 0.074*"report" + 0.048*"fear" + 0.044*"set"
15, 0.016709083691239357, 0.084*"council" + 0.074*"face" + 0.063*"help" + 0.061*"woman" + 0.056*"farmer"
16, 0.016709083691239357, 0.176*"polic" + 0.080*"water" + 0.048*"seek" + 0.035*"offic" + 0.034*"brisban"
17, 0.016709083691239357, 0.046*"case" + 0.043*"return" + 0.041*"trial" + 0.037*"countri" + 0.037*"releas"
18, 0.016709083691239357, 0.126*"court" + 0.066*"qld" + 0.044*"law" + 0.038*"drug" + 0.037*"big"
19, 0.016709083691239357, 0.096*"chang" + 0.055*"coast" + 0.053*"war" + 0.039*"east" + 0.037*"gold"
```

Model Testing:

Below is the list of the tuple for the bow_vector for unseen text given as an input for testing our model.

```
[(140, 1), (171, 1), (279, 1), (483, 1)]
```

To test we applied the bow-vector to the lda model and from the resulted output we can see that
The given unseen text may belong to topics: 4,9,17,18.
Topic 4: shows the words australia, crash, public which is similar to the above results obtained
in the model evaluation.

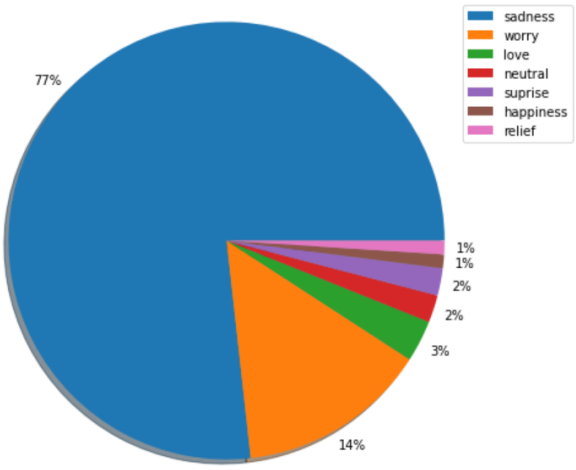
```
18, 0.4099999964237213, 0.127*"court" + 0.066*"qld" + 0.045*"law" + 0.038*"drug" + 0.037*"big"
9, 0.20999999344348907, 0.058*"call" + 0.053*"warn" + 0.040*"push" + 0.039*"deal" + 0.039*"state"
17, 0.20999999344348907, 0.046*"case" + 0.043*"return" + 0.041*"trial" + 0.037*"countri" + 0.036*"releas"
4, 0.010000000707805157, 0.090*"australia" + 0.058*"hous" + 0.051*"crash" + 0.048*"continu" + 0.037*"public"
```

2.Twitter dataset Emotional Analysis:-

As we trained the model for LSTM and ROBERT, please find the emotions of few sentences and accuracy.

LSTM:-

```
#testing the model with the given sentence
result =get_sentiment(model,"The pain my heart feels is just too much for it to bear. Nothing eases this pain. I can't hold myself back. I reall
plot_result(result)
```



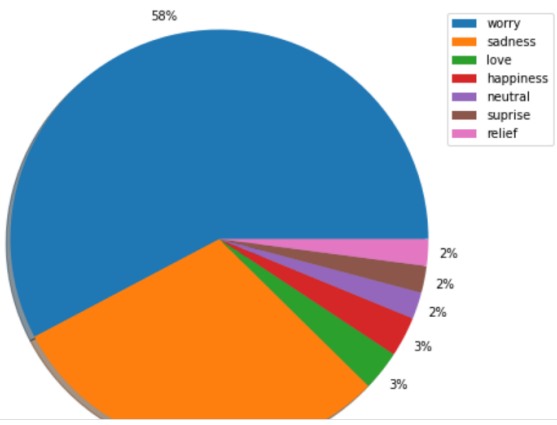
✓ 0s completed at 21:42

```
#ACCURACY OF THE MODEL

from sklearn.metrics import accuracy_score

print(accuracy_score(Y_test_pad, model.predict(X_test_pad)))
```

```
0.89
```

Robert:-

```
#testing the model with a given sentence same as used above
result =get_sentiment(model_lstm_gwe,"The pain my heart feels is just too much for it to bear. Nothing eases this pain. I can't hold myself back. I r
plot_result(result)
```



```
#ACCURACY OF THE MODEL

from sklearn.metrics import

print(accuracy_score(ytest, model.predict(xtest)))
```

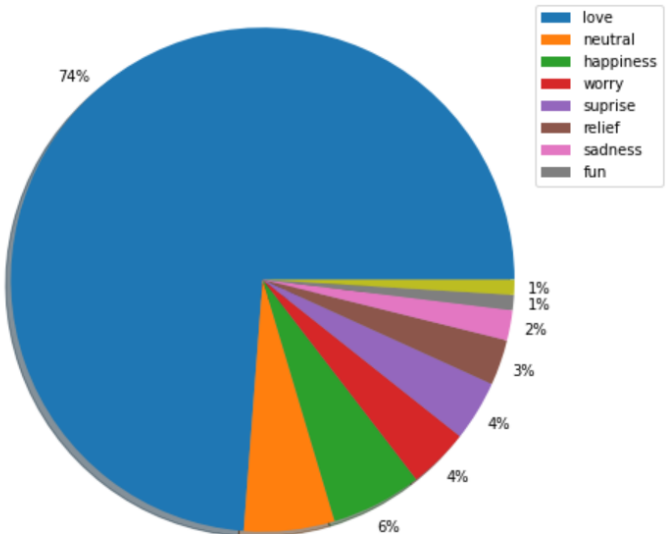
LogisticRegression: model

sklearn.linear_model._logistic.LogisticRegression instance

```
0.8623853211009175
```

Additionally we tried ALbert model

```
#testing of the model with one of the same sentence that is used above.
result =get_sentiment2(albert,"Had an absolutely brilliant day ðŸ˜ loved seeing an old friend and reminiscing")
plot_result(result)
```




```
[97] #ACCURACY OF THE MODEL

     from sklearn.metrics import accuracy_score

     print(accuracy_score(Y_test_t, model.predict(X_test_t)))
```

```
0.79
```

Project Management:

Implementation status report:

▪ Work completed for million news headline dataset:

1. Reading and cleaning data,
2. Pre-processing and feature engineering
3. Visualizing the features and interpretation of observations
4. Text classification

5. Sentiment Analysis
6. Topic Modeling and Clustering using bag-of words and tf-idf
7. Model evaluation and testing

• Description:

After reading and cleaning the data, we performed pre-processing of the data to understand about the shape, data types etc. Since the data had only two features headline_text and publish_date, we extracted the date, month, year, word_count, char_count, mean_word_length, punctuation_count, and stop_word_count.

Later the extracted features are visualized by plotting graphs. As we are dealing with a text dataset we performed text classification using the zero-shot hugging face classifier by pre-defining the labels and then performed sentiment analysis by defining the sentiment labels and applied the sequence and the sentiment-labels to the classifier. The output helps in identifying how the sequence is classified based on the classifier score, which is either positive or negative. We then performed topic modeling for the pre-processed documents are then converted into vectors using bag-of-words and tf-idf and then evaluated the models using lda for both the bag-of-words and tf-idf and then texted the unseen text and predicted the output.

• Responsibility and Contributions:

Team member	Responsibility (Tasks)	Contributions(%)
Dhathri Gundum	Research related to project, Coding: visualizing features of data and interpreting results, topic modeling using bag-of-words and tf-idf, model evaluation and testing for headlines dataset and part of the documentation	34
Sri Harsha Swaraj Nadendla	Research related to the project, coding for feature extraction, **sentiment analysis for headlines dataset**, emotion sentiment analysis for twitter dataset and state of art and documentation and project demo video.	33
Sai Dev Prakash Janapareddi	Research related to the project, Coding reading and cleaning the data and preprocessing data, text classification for headlines dataset, **emotion sentiment analysis** for headline and twitter	33

	dataset and state of art and documentation and video.	

Issues/Concerns:

Our concerns and issues while doing the project were not having the entire knowledge on the subject and not knowing what we wanted to do for our text data. With guidance from the professor we were able to understand a few and implement them by doing ample research and by discussing with team mates.

Project Final Project Video Link:-

https://drive.google.com/drive/folders/1B54l9jKfvOp3a6a2HbsaQh-8FtgW2G2A?usp=sharing

References/Bibliography:

https://www.kaggle.com/therohk/million-headlines

https://arxiv.org/pdf/2103.14465.pdf

https://blog.netcetera.com/zero-shot-text-classification-13c5236edcd3

https://ieeexplore.ieee.org/abstract/document/8999624

https://www.kaggle.com/ishivinal/tweet-emotions-analysis-using-lstm-glove-roberta

https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/

https://towardsdatascience.com/nlp-extracting-the-main-topics-from-your-dataset-using-lda-in-minutes-21486f5aa925

https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

https://towardsdatascience.com/how-i-used-natural-language-processing-to-extract-context-from-news-headlines-df2cf5181ca6

https://neptune.ai/blog/exploratory-data-analysis-natural-language-processing-tools

https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664

https://www.analyticsvidhya.com/blog/2021/01/sentiment-analysis-vader-or-textblob/

https://www.kaggle.com/thomaskonstantin/exploring-internet-news-headlines/notebook

https://www.kaggle.com/ashishbt08b004/topic-modeling-clustering-recommendations

https://towardsdatascience.com/zero-shot-text-classification-with-hugging-face-7f533ba83cd6

https://www.kaggle.com/rutwikvj/zero-shot-classification-with-hugging-face

https://medium.com/@pratikbarhate/latent-dirichlet-allocation-for-beginners-a-high-level-intuition-23f8a5cbad71